The temporal explorer who returns to the base*

Eleni C. Akrida[1], George B. Mertzios[2], and Paul G. Spirakis[1,3]

[1] Department of Computer Science, University of Liverpool, UK
{eleni.akrida,p.spirakis}@liverpool.ac.uk
[2] Department of Computer Science, Durham University, UK
george.mertzios@durham.ac.uk
[3] Computer Engineering and Informatics Department, University of Patras, Greece

Abstract. In this paper we study the problem of exploring a temporal graph (i.e. a graph that changes over time), in the fundamental case where the underlying static graph is a star on n vertices. The aim of the exploration problem in a temporal star is to find a temporal walk which starts at the center of the star, visits all leaves, and eventually returns back to the center. We present here a systematic study of the computational complexity of this problem, depending on the number k of time-labels that every edge is allowed to have; that is, on the number k of time points where each edge can be present in the graph. To do so, we distinguish between the decision version StarExp(k), asking whether a complete exploration of the instance exists, and the maximization version MaxStarExp(k) of the problem, asking for an exploration schedule of the greatest possible number of edges in the star. We fully characterize MaxStarExp(k) and show a dichotomy in terms of its complexity: on one hand, we show that for both $k = 2$ and $k = 3$, it can be efficiently solved in $O(n \log n)$ time; on the other hand, we show that it is APX-complete, for every $k \geq 4$ (does not admit a PTAS, unless P = NP, but admits a polynomial-time 1.582-approximation algorithm). We also partially characterize StarExp(k) in terms of complexity: we show that it can be efficiently solved in $O(n \log n)$ time for $k \in \{2, 3\}$ (as a corollary of the solution to MaxStarExp(k), for $k \in \{2, 3\}$), but is NP-complete, for every $k \geq 6$.

1 Introduction and motivation

A temporal graph is, roughly speaking, a graph that changes over time. Several networks, both modern and traditional, including social networks, transportation networks, information and communication networks, can be modeled as temporal graphs. The common characteristic in all the above examples is that the network structure, i.e. the underlying graph topology, is subject to discrete changes over time. Temporal graphs naturally model such time-varying networks using time-labels on the edges of a graph to indicate moments of existence of those

* Partially supported by the NeST initiative of the School of EEE and CS at the University of Liverpool and by the EPSRC Grants EP/P020372/1 and EP/P02002X/1.

edges, while the vertex set remains unchanged. This formalism originates in the foundational work of Kempe et al. [15].

In this work, we focus in particular on temporal graphs where the underlying graph is a star graph and we consider the problem of exploring such a temporal graph starting and finishing at the center of the star. The motivation behind this is inspired from the well known Traveling Salesperson Problem (TSP). The latter asks the following question: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin one?". In other words, given an undirected graph with edge weights where vertices represent cities and edges represent the corresponding distances, find a minimum-cost Hamiltonian cycle. However, what happens when the traveling salesperson has particular temporal constraints that need to be satisfied, e.g. (s)he can only go from city A to city B on Mondays or Tuesdays, or (s)he can only travel by train and, hence, needs to schedule his/her visit based on the train timetables? In particular, consider a traveling salesperson who, starting from his/her home town, has to visit $n-1$ other towns via train, always *returning to his/her own home town* after visiting each city. There are trains between each town and the home town only on specific times/days, possibly different for different towns, and the salesperson knows those times in advance. Can the salesperson decide whether (s)he can visit all towns and return to his/her own home town by a certain day?

Previous work. Recent years have seen a growing interest in dynamic network studies. Due to its vast applicability in many areas, the notion of temporal graphs has been studied from different perspectives under various names such as time-varying [1], evolving [10], dynamic [8, 11, 23, 24]; for a recent attempt to integrate existing models, concepts, and results from the distributed computing perspective see the survey papers [7] and the references therein. Various temporal analogues of known static graph concepts have also been studied in [2, 3, 5, 13, 17, 22].

Notably, temporal graph exploration has been studied before; Erlebach et al. [12] define the problem of computing a foremost exploration of all vertices in a temporal graph (TEXP), without the requirement of returning to the starting vertex. They show that it is NP-hard to approximate TEXP with ratio $O(n^{1-\varepsilon})$ for any $\varepsilon > 0$, and give explicit construction of graphs that need $\Theta(n^2)$ steps for TEXP. They also consider special classes of underlying graphs, such as the grid, as well as the case of random temporal graphs where edges appear in every step with independent probabilities. Michail and Spirakis [18] study a temporal analogue of TSP(1,2) where the objective is to explore the vertices of a complete directed temporal graph with edge weights from $\{1, 2\}$ with the minimum total cost. Ilcinkas et al. [14] study the exploration of constantly connected dynamic graphs on an underlying cactus graph. Bodlaender and van der Zanden [6] show that exploring temporal graphs of small pathwidth is NP-complete; they start from

the problem that we define in this paper[4], which we prove is NP-complete, and give a reduction to the problem of exploring temporal graphs of small pathwidth.

We focus here on the exploration of temporal stars, inspired by the Traveling Salesperson Problem (TSP) where the salesperson returns to his/her base after visiting every city. TSP is one of the most well-known combinatorial optimization problems, which still poses great challenges despite having been intensively studied for more than sixty years.

The model and definitions. It is generally accepted to describe a network topology using a graph, the vertices and edges of which represent the communicating entities and the communication opportunities between them, respectively. Unless otherwise stated, we denote by n and m the number of vertices and edges of the graph, respectively. We consider graphs whose edge availabilities are described by sets of positive integers (labels), one set per edge.

Definition 1 (Temporal Graph). *Let $G = (V, E)$ be a graph. A temporal graph on G is a pair (G, L), where $L : E \to 2^{\mathbb{N}}$ is a time-labeling function, called a* labeling *of G, which assigns to every edge of G a set of discrete-time labels. The labels of an edge are the* discrete time instances ("days") *at which it is available.*

More specifically, we focus on temporal graphs whose underlying graph is an undirected star, i.e. a connected graph of $m = n - 1$ edges which has $n - 1$ leaves, i.e. vertices of degree 1.

Definition 2 (Temporal Star). *A temporal star is a temporal graph (G_s, L) on a star graph $G_s = (V, E)$. Henceforth, we denote by c the center of G_s, i.e. the vertex of degree $n - 1$.*

Definition 3 (Time edge). *Let $e = \{u, v\}$ be an edge of the underlying graph of a temporal graph and consider a label $l \in L(e)$. The ordered triplet (u, v, l) is called* time edge.[5]

A basic assumption that we follow here is that when a message or an entity passes through an available link at time (day) t, then it can pass through a subsequent link only at some time (day) $t' > t$ and only at a time at which that link is available.

Definition 4 (Journey). *A* temporal path *or* journey *j from a vertex u to a vertex v ((u, v)-*journey*) is a sequence of time edges (u, u_1, l_1), (u_1, u_2, l_2), ... , (u_{k-1}, v, l_k), such that $l_i < l_{i+1}$, for each $1 \leq i \leq k - 1$. We call the last time label, l_k,* arrival time *of the journey.*

Given a temporal star (G_s, L), on the one hand we investigate the complexity of deciding whether G_s is *explorable*: we say that (G_s, L) is explorable if there

[4] A preliminary version of this paper appeared publicly in ArXiv on 12^{th} May 2018 (`https://arxiv.org/pdf/1805.04713.pdf`).

[5] Note that an undirected edge $e = \{u, v\}$ is associated with $2 \cdot |L(e)|$ time edges, namely both (u, v, l) and (v, u, l) for every $l \in L(e)$.

is a journey starting and ending at the center of G_s that visits every node of G_s. Equivalently, we say that there is an *exploration* that *visits* every node, and *explores* every edge, of G_s; an edge of G_s is explored by crossing it from the center to the leaf at some time t and then from the leaf to the center at some time $t' > t$. On the other hand, we investigate the complexity of computing an exploration schedule that explores the greatest number of edges. A (partial) exploration of a temporal star is a journey J that starts and ends at the center of G_s which visits some nodes of G_s; its size $|J|$ is the number of nodes of G_s that are visited by J, where the centre is only accounted for once even if it is visited multiple times. We, therefore, identify the following problems:

StarExp(k)

Input: A temporal star (G_s, L) such that every edge has at most k labels.
Question: Is (G_s, L) explorable?

MaxStarExp(k)

Input: A temporal star (G_s, L) such that every edge has at most k labels.
Output: A (partial) exploration of (G_s, L) of *maximum* size.

Note that the case where one edge e of the input temporal star has only one label is degenerate. Indeed, in the decision variant (i.e. StarExp(k)) we can immediately conclude that (G_s, L) is a no-instance as this edge cannot be explored; similarly, in the maximization version (i.e. MaxStarExp(k)) we can just ignore edge e for the same reason. We say that we "enter" an edge $e = \{c, v\}$ of (G_s, L) when we cross the edge from c to v at a time on which the edge is available. We say that we "exit" e when we cross it from v to c at a time on which the edge is available. Without loss of generality we can assume that, in an exploration of (G_s, L), the entry to any edge e is followed by the exit from e at the earliest possible time (after the entry). That is, if the labels of an edge e are $l_1, l_2, \ldots, l_k$ and we enter e at time l_i, we exit at time l_{i+1}. The reason is that, waiting at a leaf (instead of exiting as soon as possible) does not help in exploring more edges; we are better off returning to the center c as soon as possible.

In order to solve the problem of exploring as many edges of a temporal star as possible, we define here the Job Interval Selection Problem where each job has at most k associated intervals (JISP(k)), $k \geq 1$.

Job Interval Selection Problem - JISP(k) [21]

Input: n jobs, each described as a set of at most k intervals on the real line.
Output: A schedule that executes as many jobs as possible; to execute a job one needs to select one interval associated with the job.

Notice that every edge e with labels $l_1, l_2, \ldots, l_k$ can be seen as a job to be scheduled where the corresponding intervals are $[l_1, l_2], [l_2, l_3], \ldots, [l_{k-1}, l_k]$, hence MaxStarExp($k$) is a special case of JISP($k-1$); in the general JISP($k-1$), the intervals associated with each job are not necessarily consecutive. JISP(k) is a well-studied problem in the Scheduling community, with several known complexity

results. In particular, Spieksma [21] showed a 2-approximation for the problem, later improved to a 1.582-approximation by Chuzhoy et al. [9]. This immediately implies a 1.582-approximation algorithm for MaxStarExp(k); we use the latter to conclude on the APX-completeness[6] of MaxStarExp(k). JISP(k) was also shown [21] to be APX-hard for any $k \geq 2$, but since MaxStarExp(k) is a special case of JISP($k-1$), its hardness does not follow from the already known results. In fact, we show that MaxStarExp(3) -which is a special case of JISP(2)- is polynomially solvable.

Our contribution. In this paper we do a systematic study of the computational complexity landscape of the temporal star exploration problems StarExp(k) and MaxStarExp(k), depending on the maximum number k of labels allowed per edge. As a warm-up, we first prove in Section 2 that the maximization problem MaxStarExp(2) and MaxStarExp(3), i.e. when every edge has at most three labels per edge, can be efficiently solved in $O(n \log n)$ time; sorting the labels of the edges is the dominant part in the running time.

In Section 3 we prove that, for every $k \geq 6$, the decision problem StarExp(k) is NP-complete and, for every $k \geq 4$, the maximization problem MaxStarExp(k) is APX-hard, and thus it does not admit a Polynomial-Time Approximation Scheme (PTAS), unless P = NP. These results are proved by reductions from special cases of the satisfiability problem, namely 3SAT(3) and Max2SAT(3). The APX-hardness result is complemented by a 1.582-approximation algorithm for MaxStarExp(k) for any k, which concludes that MaxStarExp(k) is APX-complete for $k \geq 4$. This approximation algorithm carries over from an approximation for the Job Interval Selection Problem [9], which we show is generalization of MaxStarExp(k).

The table below summarizes the results presented in this paper regarding the complexity of the two studied problems, shows the clear dichotomy in the complexity of MaxStarExp(k), as well as the open problem regarding the complexity of StarExp(k) for $k \in \{4, 5\}$. The entry NP-c (resp. APX-c) denotes NP-completeness (resp. APX-completeness). Where $k = 1$, any instance of either problem is clearly a NO-instance, since one can explore no edge (i.e. by also returning to the center) with a single label:

	Maximum number of labels per edge					
	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k \geq 6$
StarExp(k)	No	$O(n \log n)$	$O(n \log n)$	?	?	NP-c
MaxStarExp(k)	No	$O(n \log n)$	$O(n \log n)$	APX-c	APX-c	APX-c

[6] APX is the complexity class of optimization problems that allow constant-factor approximation algorithms.

2 Efficient algorithm for $k \leq 3$ labels per edge

In this section we show that, when every edge has two or three labels, a maximum size exploration in (G_s, L) can be efficiently solved in $O(n \log n)$ time. Thus, clearly, the decision variation of the problem, i.e. StarExp(2) and StarExp(3), can also be solved within the same time bound. We give here the proof for $k = 3$ labels, which also covers the case of $k = 2$.

Theorem 1. MaxStarExp(3) *can be solved in* $O(n \log n)$ *time.*

Proof. We show that MaxStarExp(3) is reducible to the Interval Scheduling Maximization Problem (ISMP).

Interval Scheduling Maximization Problem (ISMP)
Input: A set of intervals, each with a start and a finish time.
Output: Find a set of non-overlapping intervals of maximum size.

Given (G_s, L) we construct a set I of at most $2(n-1)$ intervals as follows:

all edges of (G_s, L) with a single label can be ignored as they can not be explored in any exploration of (G_s, L); for every edge e of (G_s, L) with labels $l_e < l'_e$ we create a single *closed* time interval, $[l_e, l'_e]$; for every edge e of (G_s, L) with labels $l_e < l'_e < l''_e$ we create two *closed* time intervals, $[l_e, l'_e]$ and $[l'_e, l''_e]$.

We may now compute a maximum size subset I' of I of non-conflicting (i.e., disjoint) time intervals, using the greedy algorithm that can find an optimal solution for ISMP [16]. It suffices to observe that no two intervals associated with the same edge will ever be selected in I', as any two such intervals are non-disjoint; indeed, two intervals associated with the same edge e are of the form $[l_e, l'_e]$ and $[l'_e, l''_e]$, hence they overlap at the single time point l'_e.

So a maximum-size set I' of non-overlapping intervals corresponds to a maximum-size exploration of (G_s, L) (in fact, of the same size as the size of I'). Also, we may indeed solve StarExp(3) by checking whether $|I'| = n - 1$ or not. The above works in $O(n \log n)$ time [16]. □

3 Hardness for $k \geq 4$ labels per edge

In this section we show that, whenever $k \geq 6$, StarExp(k) is NP-complete. Furthermore, we show that MaxStarExp(k) is APX-hard for $k \geq 4$. Thus, in particular, MaxStarExp(k) does not admit a Polynomial-Time Approximation Scheme (PTAS), unless P = NP. In fact, due to a known polynomial-time constant-factor approximation algorithm for JISP(k) [9], it follows that MaxStarExp(k) is also APX-complete.

3.1 StarExp(k) is NP-complete for $k \geq 6$ labels per edge

We prove our NP-completeness result through a reduction from a special case of 3SAT, namely 3SAT(3), which is known to be NP-complete [19].

3SAT(3)

Input: A boolean formula in CNF with variables $x_1, x_2, \ldots, x_p$ and clauses $c_1, c_2, \ldots, c_q$, such that each clause has at most 3 literals, and each variable appears in at most 3 clauses.
Output: Decision on whether the formula is satisfiable.

Intuition and overview of the reduction: Given an instance F of 3SAT(3), we shall create an instance (G_s, L) of StarExp(k) such that F is satisfiable if and only if (G_s, L) is explorable. Henceforth, we denote by $|\tau(F)|$ the number of clauses of F that are satisfied by a truth assignment τ of F. Without loss of generality we make the following assumptions on F. Firstly, if a variable occurs only with positive (resp. only with negative) literals, then we trivially set it to *true* (resp. $false$) and remove the associated clauses. Furthermore, without loss of generality, if a variable x_i appears three times in F, we assume that it appears once as a negative literal $\neg x_i$ and two times as a positive literal x_i; otherwise we rename the negation with a new variable. Similarly, if x_i appears two times in F, then it appears once as a negative literal $\neg x_i$ and once as a positive literal x_i.

We introduce here the intuition behind the reduction. (G_s, L) will have one edge corresponding to each clause of F, and three edges (one "primary" and two "auxiliary" edges) corresponding to each variable of F. We shall assign labels in pairs to those edges so that it is possible to explore an edge only by using labels from the same pair to enter and exit the edge; for example, if an edge e is assigned the pairs of labels l_1, l_2 and l_3, l_4, with $l_1 < l_2 < l_3 < l_4$, we shall ensure that one cannot enter e with, say, label l_2 and exit with, say, label l_3. In particular, for the "primary" edge corresponding to a variable x_i we will assign to it two pairs of labels, namely $(\alpha i - \beta, \alpha i - \beta + \gamma)$ and $(\alpha i + \beta, \alpha i + \beta + \gamma)$, for some $\alpha, \beta, \gamma \in \mathbb{N}$. The first (entry,exit) pair corresponds to setting x_i to false, while the second pair corresponds to setting x_i to true. We shall choose α, β, γ so that the entry and exit from the edge using the first pair is not conflicting with the entry and exit using the second pair.

Then, to any edge corresponding to a clause c_j that contains x_i unnegated for the first time (resp. second time[7]), we shall assign an (entry, exit) pair of labels $(\alpha i - \delta, \alpha i - \delta + \varepsilon)$ (resp. $(\alpha i - \delta', \alpha i - \delta' + \varepsilon')$), choosing $\delta, \varepsilon \in \mathbb{N}$ (resp. $\delta', \varepsilon' \in \mathbb{N}$) so that $(\alpha i - \delta, \alpha i - \delta + \varepsilon)$ (resp. $(\alpha i - \delta', \alpha i - \delta' + \varepsilon')$) is in conflict with the $(\alpha i - \beta, \alpha i - \beta + \gamma)$ pair of labels of the edge corresponding to x_i, which is associated with $x_i = false$ but not in conflict with the $(\alpha i + \beta, \alpha i + \beta + \gamma)$ pair. If x_i is false in F then c_j cannot be satisfied through x_i so we should not be able to explore a corresponding edge via a pair of labels associated with x_i. If c_j contains x_i negated, we shall assign to its corresponding edge an (entry, exit) pair of labels $(\alpha i + \zeta, \alpha i + \zeta + \theta)$, choosing $\zeta, \theta \in \mathbb{N}$ so that the latter is in conflict with the $(\alpha i + \beta, \alpha i + \beta + \gamma)$ pair of labels of the edge corresponding to x_i, which is associated with $x_i = true$ but not in conflict with the $(\alpha i - \beta, \alpha i - \beta + \gamma)$ pair.

[7] We consider here the order $c_1, c_2, \ldots, c_q$ of the clauses of C; we say that x_i appears unnegated for the *first* time in some clause c_μ if $x_i \notin c_m,\ m < \mu$.

If x_i is true in F then c_j cannot be satisfied through $\neg x_i$ so we should not be able to explore a corresponding edge via a pair of labels associated with $\neg x_i$.

Finally, for every variable x_i we also introduce two additional "auxiliary" edges: the first one will be assigned the pair of labels $(\alpha i, \alpha i + \xi)$, $\xi \in \mathbb{N}$, so that it is not conflicting with any of the above pairs – the reason for introducing this first auxiliary edge is to avoid entering and exiting an edge corresponding to some variable x_i using labels from different pairs. The second auxiliary edge for variable x_i will be assigned the pair of labels $(\alpha i+\chi, \alpha i+\chi+\psi)$, $\chi, \psi \in \mathbb{N}$, so that it is not conflicting with any of the above pairs – the reason for introducing this edge is to avoid entering an edge that corresponds to some clause c_j using a label associated with some variable x_i and exiting using a label associated with a *different* variable $x_{i'}$. The reader is referred to Figure 1 for an example construction, where the specific choices of the constants $\alpha, \beta, \gamma, \delta, \varepsilon, \delta', \varepsilon', \zeta, \theta, \xi, \chi, \psi$ are $\alpha = 50, \beta = 10, \gamma = 3, \delta = 12, \varepsilon = 3, \delta' = 8, \varepsilon' = 3, \zeta = 8, \theta = 3, \xi = 1, \chi = 15, \psi = 1$.

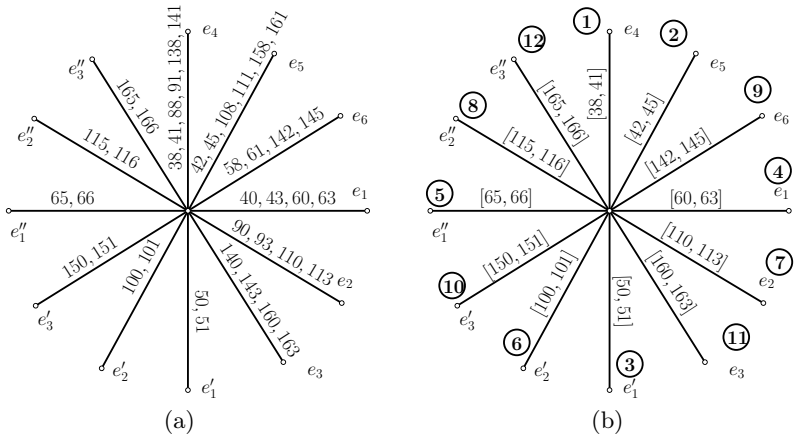

Fig. 1. The temporal star constructed for the formula $(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_3)$. Setting x_1 to true, x_2 to true and x_3 to true yields a satisfying truth assignment whose corresponding exploration is indicated in (b), where the numbers in the circles indicate the order over time of the exploration of each edge.

The following lemmas are needed for the NP-completeness proof (Theorem 2).

Lemma 1. *There exists a (partial) exploration J of (G_s, L) of maximum size which explores all $3p$ edges associated with the variables of F.*

Lemma 2. *There exists a truth assignment τ of F with $|\tau(F)| \geq \beta$ if and only if there exists a (partial) exploration J of (G_s, L) of size $|J| \geq 3p + \beta$.*

Theorem 2. StarExp(k) *is NP-complete for every $k \geq 6$.*

3.2 MaxStarExp(k) is APX-complete for $k \geq 4$ labels per edge

It can be shown that the reduction of Section 3.1 linearly preserves approximability features; this would in turn prove that MaxStarExp(k) is APX-hard for $k \geq 6$, since MAX3SAT(3), i.e. the maximization version of 3SAT(3), is APX-complete [4]. However, this leaves a gap in the complexity of the problem for $k \in \{4, 5\}$. To close this gap we instead give an *L-reduction* [20] from the Max2SAT(3) problem, i.e. an approximation preserving reduction which linearly preserves approximability features. Max2SAT(3) is known to be APX-complete [20].

MAX2SAT(3)

Input: A boolean formula in CNF with variables $x_1, x_2, \ldots, x_p$ and clauses $c_1, c_2, \ldots, c_q$, such that each clause has at most 2 literals, and each variable appears in at most 3 clauses.
Output: Maximum number of satisfiable clauses in the formula.

The reduction: Given an instance F of MAX2SAT(3) we shall create an instance (G_s, L) of MaxStarExp(k) such that F has β satisfiable clauses if and only if (G_s, L) has $\beta + 3p$ explorable edges. As previously, we assume without loss of generality that every variable appears once as a negative literal and once or twice as a positive literal.

The reduction is the same as the one presented in Section 3.1, with the edges of (G_s, L) being assigned the same labels as in the previous reduction to appropriately introduce conflicts between exploration windows of edges. The only difference in the construction is that now we start from a 2-CNF formula F (instead of a 3-CNF formula in Section 3.1). Thus every edge of (G_s, L) that corresponds to a clause of F now receives four labels instead of six, i.e. two labels for every literal that appears in the clause.

The following lemmas are needed for the APX-hardness proof (Theorem 3).

Lemma 3. *There exists a (partial) exploration J of (G_s, L) of maximum size which explores all $3p$ edges associated with the variables of F.*

Lemma 4. *There exists a truth assignment τ of F with $|\tau(F)| \geq \beta$ if and only if there exists a (partial) exploration J of (G_s, L) of size $|J| \geq 3p + \beta$.*

Theorem 3. MaxStarExp(k) *is APX-hard, for $k \geq 4$.*

Proof. Denote by $OPT_{\text{Max2SAT(3)}}(F)$ the greatest number of clauses that can be simultaneously satisfied by a truth assignment of F. The proof is done by an *L-reduction* [20] from the Max2SAT(3) problem, i.e. by an approximation preserving reduction which linearly preserves approximability features. For such a reduction, it suffices to provide a polynomial-time computable function g and two constants $\gamma, \delta > 0$ such that:

- $OPT_{\text{MaxStarExp}}((G_s, L)) \leq \gamma \cdot OPT_{\text{Max2SAT(3)}}(F)$, for any boolean formula F, and

– for any (partial) exploration J' of (G_s, L), $g(J')$ is a truth assignment for F and $OPT_{\text{MAX2SAT(3)}}(F) - |g(J')| \leq \delta(OPT_{\text{MAXSTAREXP}}((G_s, L)) - |J'|)$, where $|g(J')|$ is the number of clauses of F that are satisfied by $g(J')$.

We will prove the first condition for $\gamma = 13$. Recall that p and q are the numbers of variables and clauses of F, respectively. Note that a random truth assignment satisfies each clause of F with probability at least $\frac{1}{2}$ (if each clause had exactly 2 literals, then it would be satisfied with probability $\frac{3}{4}$, but we have to account also for single-literal clauses), and thus there exists an assignment τ that satisfies at least $\frac{q}{2}$ clauses of F. Furthermore, since every clause has at most 2 literals and every variable appears at least once, it follows that $q \geq \frac{p}{2}$. Therefore $OPT_{\text{MAX2SAT(3)}}(F) \geq \frac{q}{2} \geq \frac{p}{4}$, and thus $p \leq 4 \cdot OPT_{\text{MAX2SAT(3)}}(F)$. Now Lemma 4 implies that:

$$\begin{aligned} OPT_{\text{MAXSTAREXP}}((G_s, L)) &= 3p + OPT_{MAX2SAT(3)}(F) \\ &\leq 3 \cdot 4 \cdot OPT_{MAX2SAT(3)}(F) + OPT_{MAX2SAT(3)}(F) \\ &= 13 \cdot OPT_{\text{MAX2SAT(3)}}(F) \end{aligned}$$

To prove the second condition for $\delta = 1$, consider an arbitrary partial exploration J' of $G_s(L)$ of maximum size. In the proof of Lemma 4, we describe how one can start from any such J' and construct in polynomial time a truth assignment $g(J') = \tau$ that satisfies at least $OPT_{\text{MAXSTAREXP}}((G_s, L)) - 3p$ clauses of F, i.e. $|g(J')| = |\tau(F)| \geq |J'| - 3p$. Then:

$$\begin{aligned} OPT_{\text{MAX2SAT(3)}}(F) - |g(J')| &\leq OPT_{\text{MAX2SAT(3)}}(F) - |J'| + 3p \\ &= OPT_{\text{MAXSTAREXP}}((G_s, L)) - 3p - |J'| + 3p \\ &= OPT_{\text{MAXSTAREXP}}((G_s, L)) - |J'| \end{aligned}$$

This completes the proof of the theorem. □

Corollary 1. MAXSTAREXP(k) *is APX-complete, for* $k \geq 4$.

Now we prove a correlation between the inapproximability bounds for the MAXSTAREXP(k) problem and MAX2SAT(3), as a result of the L-reduction presented in the proof of Theorem 3. Note that, since MAX2SAT(3) is APX-hard [4], there exists a constant $\varepsilon_0 > 0$ such that there exists no polynomial-time constant-factor approximation algorithm for MAX2SAT(3) with approximation ratio greater than $(1 - \varepsilon_0)$, unless P = NP.

Theorem 4. *Let* $\varepsilon_0 > 0$ *be the constant such that, unless P = NP, there exists no polynomial-time constant-factor approximation algorithm for* MAX2SAT(3) *with approximation ratio greater than* $(1 - \varepsilon_0)$. *Then, unless P = NP, there exists no polynomial-time constant-factor approximation algorithm for* MAXSTAREXP(k) *with approximation ratio greater than* $(1 - \frac{\varepsilon_0}{13})$.

Proof. Let $\varepsilon > 0$ be a constant such that there exists a polynomial-time approximation algorithm $\mathcal{A}$ for MaxStarExp(k) with ratio $(1-\varepsilon)$. Let F be an instance of MAX2SAT(3) with p variables and q clauses. We construct the instance (G_s, L) of MaxStarExp(k) corresponding to F, as described in the L-reduction (see Theorem 3). Then we apply the approximation algorithm $\mathcal{A}$ to (G_s, L), which returns a (partial) exploration J. Note that $|J| \geq (1-\varepsilon) \cdot OPT_{\text{MaxStarExp}}$. We construct from J in polynomial time a truth assignment τ; we denote by $|\tau|$ the number of clauses in F that are satisfied by the truth assignment τ. It now follows from the proof of Theorem 3 that:

$$\begin{aligned} OPT_{\text{Max2SAT(3)}}(F) - |\tau| &\leq OPT_{\text{MaxStarExp}}((G_s, L)) - |J| \\ &\leq 13\varepsilon \cdot OPT_{\text{Max2SAT(3)}}(F) \end{aligned}$$

Therefore $|\tau| \geq (1-13\varepsilon) \cdot OPT_{\text{Max2SAT(3)}}(F)$. That is, using algorithm $\mathcal{A}$, we can devise a polynomial-time algorithm for MAX2SAT(3) with approximation ratio $(1-13\varepsilon)$. Therefore, due to the assumptions of the theorem it follows that $\varepsilon \geq \frac{\varepsilon_0}{13}$, unless P = NP. This completes the proof of the theorem. □

Note that we have fully characterized MaxStarExp(k) in terms of complexity, for all values of $k \in \mathbb{N}$. However, the reduction that shows APX-hardness for MaxStarExp(k) cannot be employed to show NP-hardness of the decision version StarExp(k), since the decision problem 2SAT is polynomially solvable.

Open Problem *What is the complexity of* StarExp(k)*, for* $k \in \{4, 5\}$*?*

References

1. E. Aaron, D. Krizanc, and E. Meyerson. DMVP: foremost waypoint coverage of time-varying graphs. In *International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, pages 29–41, 2014.
2. E. C. Akrida, L.Gasieniec, G. B. Mertzios, and P. G. Spirakis. The complexity of optimal design of temporally connected graphs. *Theory of Computing Systems*, 61(3):907–944, 2017.
3. E. C. Akrida, G. Mertzios, P. G. Spirakis, and V. Zamaraev. Temporal vertex covers and sliding time windows. In *International Colloquium on Automata, Languages and Programming (ICALP)*, 2018.
4. G. Ausiello, M. Protasi, A. Marchetti-Spaccamela, G. Gambosi, P. Crescenzi, and V. Kann. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties.* Springer-Verlag, 1st edition, 1999.
5. S. Biswas, A. Ganguly, and R. Shah. Restricted shortest path in temporal graphs. In *Database and Expert Systems Applications (DEXA)*, 2015.
6. H. L. Bodlaender and T. C. van der Zanden. On exploring temporal graphs of small pathwidth. *CoRR*, abs/1807.11869, 2018.
7. A. Casteigts and P. Flocchini. Deterministic Algorithms in Dynamic Networks: Formal Models and Metrics. Technical report, Defence R&D Canada, April 2013.
8. T.-H. Hubert Chan and Li Ning. Fast convergence for consensus in dynamic networks. *ACM Trans. Algorithms*, 10, 2014.

9. J. Chuzhoy, R. Ostrovsky, and Y. Rabani. Approximation algorithms for the job interval selection problem and related scheduling problems. *Mathematics of Operations Research*, pages 730–738, 2006.
10. A. E. F. Clementi, C. Macci, A. Monti, F. Pasquale, and R. Silvestri. Flooding time of edge-markovian evolving graphs. *SIAM Journal on Discrete Mathematics (SIDMA)*, 24(4):1694–1712, 2010.
11. C. Demetrescu and G. F. Italiano. Algorithmic techniques for maintaining shortest routes in dynamic networks. *Electronic Notes in Theoretical Computer Science*, 171, 2007.
12. T. Erlebach, M. Hoffmann, and Frank Kammer. On temporal graph exploration. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 444–455, 2015.
13. A.-S. Himmel, H. Molter, R. Niedermeier, and M. Sorge. Adapting the bron-kerbosch algorithm for enumerating maximal cliques in temporal graphs. *Social Network Analysis and Mining*, 7(1):35:1–35:16, 2017.
14. D. Ilcinkas, R. Klasing, and A. M. Wade. Exploration of constantly connected dynamic graphs based on cactuses. In *International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, 2014.
15. D. Kempe, J. M. Kleinberg, and A. Kumar. Connectivity and inference problems for temporal networks. In *ACM symposium on Theory of computing (STOC)*, pages 504–513, 2000.
16. J. Kleinberg and E. Tardos. *Algorithm Design*. Addison-Wesley Longman, 2005.
17. G. B. Mertzios, O. Michail, I. Chatzigiannakis, and P. G. Spirakis. Temporal network optimization subject to connectivity constraints. In *International Colloquium on Automata, Languages and Programming (ICALP)*, pages 657–668, 2013.
18. O. Michail and P. G. Spirakis. Traveling salesman problems in temporal graphs. *Theoretical Computer Science*, 634:1–23, 2016.
19. C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, 1982.
20. C. H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43(3):425–440, 1991.
21. F. C. R. Spieksma. On the approximability of an interval scheduling problem. *Journal of Scheduling*, pages 215–227, 1999.
22. T. Viard, M. Latapy, and C. Magnien. Computing maximal cliques in link streams. *Theoretical Computer Science*, 609:245–252, 2016.
23. D. Wagner, T. Willhalm, and C. D. Zaroliagis. Dynamic shortest paths containers. *Electronic Notes in Theoretical Computer Science*, 92, 2004.
24. H. Zhuang, Y. Sun, J. Tang, J. Zhang, and X. Sun. Influence maximization in dynamic social networks. In *International Conference on Data Mining*, 2013.